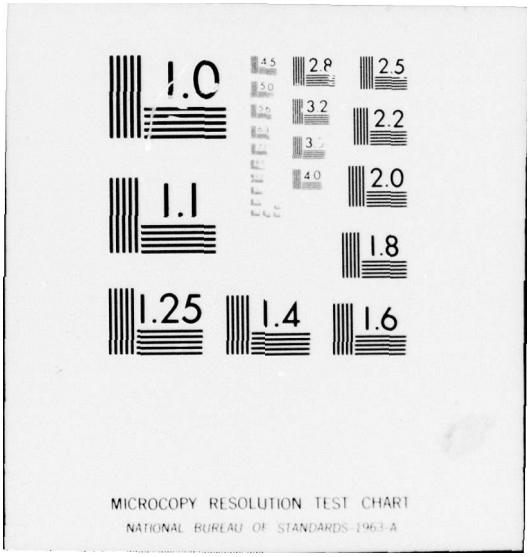
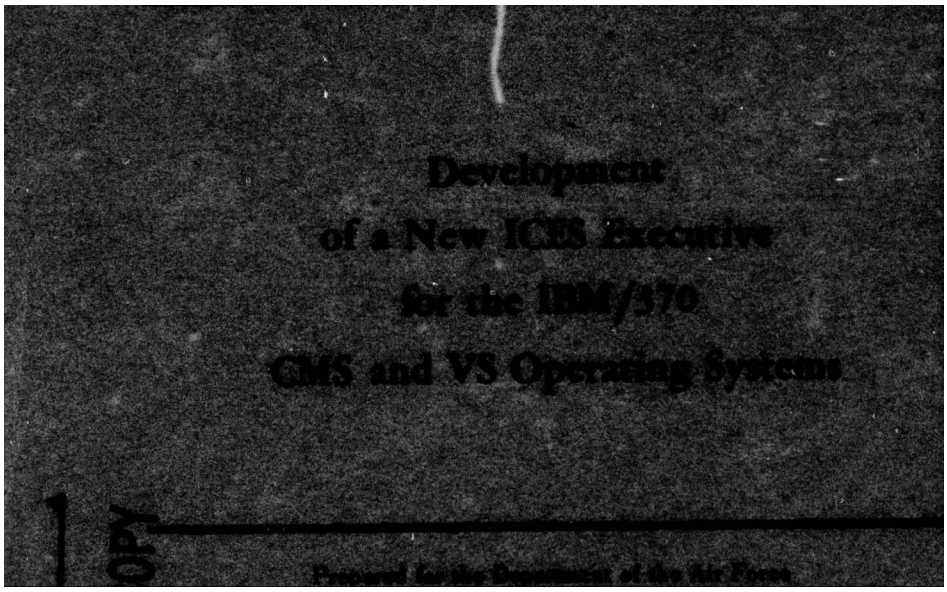

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A


Development
of a New ICES Executive
for the IBM/370
GMS and VS Operating Systems
Prepared for the Department of the Air Force

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology, with the support of the Department of the Air Force under Contract F19628-76-C-0002.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Reginald L. Lassiter, Jr., Col., USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

DEVELOPMENT OF A NEW ICES EXECUTIVE FOR THE IBM/370 CMS AND VS OPERATING SYSTEMS

B. SCHUMACKER
Consultant

Group 73

TECHNICAL NOTE 1977-1

11 JANUARY 1977

LEXINGTON MASSACHUSETTS

ABSTRACT

A new ICES executive was developed for the IBM/370 to provide more flexibility for engineering use via on-line and background computing environments and also to provide ease of maintainability and improve further development. Input can be prepared and checked (syntactically and graphically) in an on-line mode, small analyses run and checked on-line, large analyses run in batch mode, and results checked graphically on-line.

Two new versions of ICES exist: an on-line (CMS) version and a batch (VS) version. The source, except for a few programs, is independent of the version. This report describes the changes made, the reasons for the changes, new commands developed for on-line use, and performance comparisons with the previous M.I.T. version of ICES.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION AND BACKGROUND

ICES was originally developed by the M.I.T. Civil Engineering Systems Laboratory and made available to the public in 1967.[1] That original version was developed for an IBM System 360 computer and ran on both stand-alone batch (PCP) and multi-tasking batch (MVT) of OS/360. ICES and STRUDL were implemented at M.I.T. Lincoln Laboratory in the late 1960s as batch systems on the S/360.

When the Model 67 of the S/360 was installed at the M.I.T. campus, an interactive version of ICES was developed to run under CP-67/CMS.[2] This version included many of the changes made later at Lincoln Laboratory for S/370 CMS-ICES and which are the subject of this report. The M.I.T. CP-67/CMS version was developed as an experimental version and hence was not made available to the public. However, an object version of CP-67/CMS ICES was obtained by Lincoln Laboratory and used for small STRUDL runs and program development until the IBM 360 computer was replaced with the IBM 370 computer. Due to differences in CMS under the two systems, the 360 CMS version of ICES would not operate under 370 CMS. Unfortunately, by this time the interactive version at the M.I.T. campus was no longer being used, and all source for the CP-67/CMS version of ICES had been lost.

This non-availability of source presented several problems; availability of source when going out for bids on a new computer, and availability of source for maintaining and developing those systems used for contract jobs.

It was desired to have STRUDL available for general use in both interactive and batch environments, with commonality of input files. VM/370 provides this capability since CMS files are able to be used as input streams to VS/370. From an engineering user standpoint, the availability of STRUDL in both environments is highly desirable. The CMS (interactive) environment can be used for command syntax checking and for relatively small analyses (on the order of up to 3 CPU minutes), while the batch (VS) environment can be used for larger runs (greater than 3 CPU minutes). The CMS environment is also useful for the maintenance and development of subsystems such as STRUDL.

For these reasons, the decision was made to develop a CMS/370 version of ICES, working from the current V1M3 source distributed by the ICES Users Group and the unpublished write-up of the changes made to implement CP-67/CMS ICES.[3] Compatibility between the source files, the source listings, and the object (compiled source) files was required.

The STRUDL subsystem had been implemented at Lincoln Laboratory under 360/CMS and subroutine listings and source object files were available from that conversion. In that effort the source subroutines had been compiled under G-Level Fortran and had removed source program incompatibilities between E-Level and G-Level Fortran, most notably:

a. The rules for branching into DO Loops must be adhered to in Fortran G, whereas they do not have to be followed in Fortran E. There were a number of STRUDL programs in which illegal branches into DO loops were made.

b. A variable can be equivalenced to an element of a dimensioned array beyond the limit of its dimensioned length in Fortran E; in Fortran G, this results in an error during compilation. Several STRUDL programs required a change to eliminate this.

Even though the 370/CMS version of STRUDL was compiled with Fortran G, the ICETRAN precompiler was not changed to permit G-Level Fortran statements. G-Level statements may be added after COMMON resolution statements by placing a *P* in card column 1 which causes the precompiler to bypass that statement. COMMON resolution statements can be either DYNAMIC ARRAY or DOUBLE PRECISION statements as well as COMMON statements. This may only be done for statements which do not contain dynamic array references.

The 370/CMS STRUDL implementation, then, involved the generation of all the modules and the recreation of the CDBs in the ICES system data set DD5. Additional work was done on 370/CMS STRUDL, however, to update, expand, and debug the finite element, dynamics, and substructuring capabilities.[5,6,7]

After the 370/CMS-ICES (hereafter called CMS-ICES) implementation was completed, a similar version was developed for VS. Here the objective was to have a batch version which was exactly the same as the interactive version at the subsystem level and which had minimal variations in programs at the ICES executive level. This latter requirement was achieved with the variations occurring in the I/O macros and the module generation procedure.

This report describes the changes which were made to the ICES System to develop a time sharing version for 370-CMS and a 370 VS version. In this report, the following nomenclature is used:

1. OS-ICES: the version of ICES written for the IBM System 360 to operate under the OS/360 or OS/370 Operating System.

2. CMS-ICES: the version of ICES developed at Lincoln Laboratory to operate under VM/370 (Virtual Machine Facility for the IBM S/370) using the CMS (Conversational Monitor System) subset of VM/370.

3. VS-ICES: the version of ICES developed at Lincoln Laboratory to operate under the VS (Virtual Memory System) operating system for the IBM S/370. Versions for both VS1 and VS2 were developed, and VS-ICES refers to either of them.

II. *ICES* DEVELOPMENT

A. *OS TO CMS CONVERSION*

The differences between OS and CMS-ICES are described in the following sections.

1. Program Management and Dynamic Loading

The OS-ICES approach to dynamic loading is retained, i.e., single entry load modules are used with extra entry points being obtained with the use of aliases and a start-up routine at the beginning of each module to search a table for the correct entry point for the alias. However, the CMS modules may have unresolved external references resolvable at load time, system COMMON being one of these.

A directory of module and alias names is necessary so that the ICES program management routines may know the actual module to load for any requested entry point. This directory information is placed in a file of type BLDL that is read by a root program BLDL. This replaces the OS BLDL macro which returns the same information. Since CMS dynamic loading can be from a TXTLIB (a collection of TEXT files with a directory), a notation can be made in the BLDL file indicating the TXTLIB or TXTLIBs to be searched by CMS to find the required module. If the BLDL files indicate that one or more TXTLIBs are to be searched, the BLDL routine issues a GLOBAL TXTLIB command for those mentioned.

Since the BLDL information must reside in memory and since this information for all subsystems would take up an unwarranted amount of space, the BLDL routine is arranged to read a system directory (ICES BLDL) and a subsystem directory (*subsystemname* BLDL). When a new subsystem is initiated, the BLDL routine overlays the old subsystem BLDL with the new, the subsystem name being obtained from the first 8 bytes of system COMMON.

The LINKEDIT routine is used to create the single entry load modules required. They are in fact normal CMS TEXT files that can be used by the CMS dynamic loading routines. The LINKEDIT routine is described in the section on load module generation.

Since the implementation of the CMS LOAD macro would not reuse memory space correctly, the LOAD was replaced by a combination GETMAIN and INCLUDE; the DELETE was thus replaced by a FREEMAIN.

Overlay structuring of load modules in CMS-ICES is not supported because of the complexity of the problem. The root is thus not overlain, leading to greater efficiency in the root routines but cutting down the user area available to subsystem programs and data.

2. COMMON References

The whole method of handling a systemwide COMMON is changed for CMS-ICES. In OS, COMMON can only be placed within load modules, there being no provision for the systemwide COMMON required by ICES. Thus, part of the starting up routine in all OS-ICES modules searches all the routines in the modules that contain COMMON references and replaces the references in their prologues with the actual address provided by the ICES root.

In CMS-ICES the load modules may have references resolved to a systemwide COMMON at load time, the only requirement being that the ICES root must place the correct COMMON address in the CMS loader reference table. This latter step is necessary because a combined system and subsystem COMMON is reobtained every time a new subsystem is initialized. The root routines always obtain the COMMON address from the entry point QQCOMADR which is updated by INAL, the COMMON obtaining routine.

One problem, however, is that ICES unfortunately makes frequent use of the fact that E-Level FORTRAN keeps the COMMON address in register 4, while G-Level usually keeps the address of the previous save area in this register. Nearly all the interface routines

included in modules from the ICES function library assume the COMMON address is in register 4 and many of the routines they call in the root also assume this. Therefore, most of the function library routines have had to be altered to obtain the COMMON address from an entry point QQCOMADR included in the load module starting routine STARTMOD. It is also necessary to make a number of routines return to the interface routines to have register 4 reset before returning to the G-Level routines in the module. This reduces the efficiency of the dynamic data routines in the root as each return is delayed. In addition, debugging is complicated because the return indicated when a dynamic array error is obtained is in an interface routine instead of the real calling routine. The real return will be printed in one of the two words which follow the current error message return word.

CMS-ICES uses bytes 296 to 319 of system common for passing I/O indicators to the ICESIO routine.

3. Primary Memory Data Management

Since CMS has its own separate area for its work space, it is only necessary to ensure that ICES leaves enough space in high address memory for any possible use by CMS. FREELOWE is the CMS nucleus indicator that indicates the lowest high core location used by CMS. If CMS requires more space, it may lower FREELOWE. The starting up routine for CMS-ICES (STRTICES) ensures that CMS has at least two free pages for its work.

The only other intruder into the user area is FORTRAN which requires buffer space for datasets. CMS-ICES provides an interface for datasets 5 and 6 to allow interactive manipulation of input/output and buffer space for these is provided within the root. However, if a subsystem user uses other datasets, FORTRAN will obtain this space with GETMAINs from the user area. There is no reasonable way to control this and fragmentation of storage may result.

4. Secondary Storage Data Management

All OS disk management macros are replaced with the equivalent CMS macros. There are two important differences between these OS and CMS routines. In OS the record number is the relative record number and thus the first record is referred to as 0. In CMS this is record 1. In OS all records in a file must be initialized, and when all have been used, the initialization must be extended. In CMS all records on the user's disk are initialized and records are only allotted to a file when the record is actually going to be written. Thus, the concept of initialization is superfluous. This section of coding is thus removed and as far as CMS-ICES is concerned, 12448 records have been initialized for a file, this being the maximum that can be handled in the ICES track directory.

The four ICES datasets are named DD1 ICES, DD2 ICES, DD3 ICES, and DD4 ICES for CMS-ICES. The names but not the filetypes may be changed by using the ICES renaming facility by issuing a call to QQDSNC.

CALL QQDSNC (*dataset number, new name*)

5. ICES Load Module Generation

Load module generation is controlled by the CMS EXEC procedure MAKEMOD.

Basically, an assembly language routine STARTMOD is created for each module using the entry point information in the subsystem BLDL file. The entry names are placed in a table in the STARTMOD routine, and during the LINKEDITing procedure the addresses of the entry points are also placed in the table. The ICES program management routines can search this table for the required entry point address.

The assembled routine and the other routines required in the module are then LINKEDITed together to form the module, a CMS TEXT file. The final step is the automatic updating of the BLDL file entry for the module with the module size.

In CMS-ICES, load modules are relocatable CMS TEXT files produced by the LINKEDIT procedure during load module generation.

B. CMS TO VS CONVERSION

Conversion to VS was oriented toward keeping as many programs as possible the same as in CMS and improving disk operations by increasing the size of the disk blocks. On CMS the disk blocks were kept at 800 since that is the size used by CMS for its blocking. For VS, a block size of 4000 was chosen.

1. ICES Load Module Generation

The module generation procedure for VS was designed to use CMS text decks (i.e., compiles would be done on CMS) and non-overlay modules. Texts for one subsystem are stored in one partitioned dataset, and the VS utility IEBUPDTE is used to add/replace texts.

Since VS does not have anything comparable to a loader table, a different method for handling common resolution at module load time had to be found. To this end, a new OSSETGEN program was written which would find all adcons to COMMON in each text to be included in a module and would then generate code for program SETUP, which program would be executed when the module was given control after load time. The code generated for SETUP would cause each adcon to be readjusted from linkage editor generated load module COMMON address to the global COMMON address for that subsystem.

The lengths of COMMON in each text (with COMMON) are changed to 4 by program NIXCO which is invoked at the end of the PRECOMP procedure in CMS.

All other changes in the VS conversion were made to programs which had CMS macros in them, primarily I/O macros.

2. Difference Between VS1 ICES and VS2 ICES

The difference in the primary memory management methods between VS1 and VS2 has caused two different versions of OSFINCH2 to be developed. VS2 allocates primary memory in the manner in which ICES assumed for its QQQICEX2 scheme, namely, GETMAINs from high address down and LOADs from low address up. VS1, however, allocates everything (except LINKs) from high address down, and in so doing will increase the chances for memory fragmentation to such an extent that problems run on VS1 will take about 50% more CPU time and 100% more disk I/O operations than that required on VS2. This degradation was caused by fragmentation which in turn caused increased numbers of of reorgs at all levels.

Since this kind of degradation was intolerable, a modified version of OSFINCH2 was written which, by the use of GETMAINs and FREEMAINs, forces a LOAD to be made in the lowest address available. With this change, the two versions of ICES, for VS1 and VS2, have comparable running times.

C. TIME COMPARISON BETWEEN *OS-ICES* AND *VS-ICES*

Two STRUDL jobs were run on OS-ICES and VS-ICES under various configurations to compare performance on the two systems. The first STRUDL job was a dynamic analysis problem with 1,166 degrees of freedom and determines by iteration the first 5 eigenvalues. This problem had 213 joints, 136 members, and 812 elements of which 384 were CSTG, 384 were CPT, and 44 were SBCT. The maximum half-bandwidth started at 128 and was reduced to 30.

The second STRUDL job was a stiffness analysis of the same structure in the first job with five additional members and five additional joints. There were 25 loading conditions imposed upon the structure. The maximum half-bandwidth started at 130 and was reduced to 34.

The differences in the ICES systems were:

1. The OS system was the non-overlay root with modifications to permit a blocksize of 6400 on DD4 and otherwise equivalent to V1M2 version of the root, i.e., it did not have facility for 50 pools (only 20) or for the REORG command. The ICES root used here was the experimental field test version of the ECI V1M2 executive before release of their proprietary version. The STRUDL system for OS was the standard Fortran E version of STRUDL and had a pool size of 350K.

2. The VS system was the new non-overlay root with a blocksize of 4000 on DD4 and otherwise equivalent to V1M4 version of the root, i.e., up to 50 pools and the REORG command were available. The STRUDL system for VS was the Fortran G compiled version with no overlay modules and had a pool size of 80K.

Each STRUDL job was run once on the OS version in regions of 2048K and 4096K and was run several times on the VS version with different values for the REORG parameter and different region sizes.

The items measured were CPU time, number of direct access I/O operations, number of paging operations, and number of reorgs for each level of reorganization. These statistics are given in Tables II.C.1 and II.C.2.

It can be seen from these tables that the VS version with REORG 1 gives greatly improved performance over the OS version. The results also illustrate the effect of region size on system performance. It should be noted that all runs had a cut-off time of 30 CPU minutes.

TABLE II.C.1

STATISTICS FOR RUNS OF STRUDL TEST 1

(IBM 370/168)

Version	Region	Number	REORG Meaning	CPU Time	Direct Access I/O	Paging I/O	REORG Counts 0	1	2	3	4	5
OS	2M	N/A		18:14	151,686	23	0	34	7	128		
VS	2M	1	P1 P2 D1 D2 D3 D4	9:19	16,423	28	8	0	38	5	1	0
		11	D1 D2 P1 D3 P2 D4	9:24	17,997	51	58	12	1	0	0	0
		2	P1 D1 P2 D2 D3 D4	29:21	421,652	6	9	207	1	7	237	0
		3	P1 D1 D2 P2 D3 D4	29:23	418,888	73	7	135	8	0	237	0
		8	D1 D2 D3 P1 P2 D4	Cancel[1]	458,795	19	151	23	649	0	0	0
	1M	11	D1 D2 P1 D3 P2 D4	Cancel[2]	471,103	0	576	115	2	1499	0	0
		2	P1 D1 P2 D2 D3 D4	Cancel[2]	468,531	28	8	293	0	50	1041	0
	768K	1	P1 P2 D1 D2 D3 D4	Cancel[2]	446,181	714	19	1	375	455	2381	1
		3	P1 D1 D2 P2 D3 D4	Cancel[2]	471,519	5	210	966	449	0	2380	1
	4M	1	P1 P2 D1 D2 D3 D4	9:36	3,166	56,386	2	0	5	0	0	0
		11	D1 D2 P1 D3 P2 D4	9:51	4,009	73,197	5	1	0	0	0	0
		3	P1 D1 D2 P2 D3 D4	9:25	3,155	16,773	2	5	0	0	0	0
		8	D1 D2 D3 P1 P2 D4	9:46	4,042	29,642	5	1	0	0	0	0
OS	4M	N/A		8:41	5,591	13,100	0	3	1	0		

[1] working on 5th eigenvalue at cancel (due to 30 min. time)

[2] working on 4th eigenvalue at cancel (due to 30 min. time)

TABLE II.C.2

STATISTICS FOR RUN OF STRUDL TEST 2

(IBM 370/168)

Version	Region	REORG Number	REORG Meaning	CPU Time	Direct Access I/O	Paging I/O	REORG Counts 0	1	2	3	4	5
OS	2M	N/A		8:02	16,741	5	0	62	6	0		
VS	2M	1	P1 P2 D1 D2 D3 D4	7:35	13,197	1,625	2	0	58	5	0	0
		11	D1 D2 P1 D3 P2 D4	8:17	20,845	2,215	107	24	1	0	0	0
		3	P1 D1 D2 P2 D3 D4	7:47	13,212	32,634	2	58	5	0	0	0
		8	D1 D2 D3 P1 P2 D4	7:15	20,768	6,678	107	24	0	1	0	0
	4M	11	D1 D2 P1 D3 P2 D4	7:27	6,063	48,549	13	1	0	0	0	0
		3	P1 D1 D2 P2 D3 D4	7:35	6,042	54,177	2	14	1	0	0	0
OS	4M	N/A		7:21	7,599	26,481	0	10	1	0		

D. NEW USER COMMANDS

1. The IN and OUT Commands

There are two CMS-ICES System commands for controlling from where input is to come (keyboard, tape or disk file) and where output is to be directed (typewriter, tape or disk file, printer, or any combination).

```
IN    { TYPE                                                    }
      { FILE     ('filename')('filetype')('filemode')           }
      { DISK     ('filename')('filetype')('filemode')           }
      { MT       ('tapenumber')                                 }
```

Input can come from the keyboard (TYPE), or from a CMS file (FILE or DISK), or from a mag tape (MT). If a disk file is to be used, *filemode*, *filetype* and *filemode*, or *filename* and *filetype* and *filemode* need not be specified. In these cases, the default is INPUT for *filetype* and ICES for *filename*. The *tapenumber* in the mag tape option specifies the CMS tape number. The default is 1, for tape 181.

```
OUT     TYPE
      { NOTYPE                                                  }
      { PRINTER                                                 }
      { FILE     ('filename')('filetype')('filemode')           }
      { DISK     ('filename')('filetype')('filemode')           }
        MT       ('tapenumber')
```

Output can be directed to any or all of the three devices: typewriter, printer, and disk file, or typewriter, printer, and mag tape. NOTYPE must be specified to get no output. If a disk file is to be used, *filemode*, *filetype* and *filemode*, or *filename* and *filetype* and *filemode* need not be specified. In these cases, the default is

OUTPUT for *filetype*, ICES for *filename*, and A for *filemode*. If a file of *filename*, *filetype* (either specified or default) already exists, it will be added too. The default *tapenumber* is 1, again for tape 181.

Note that these commands are valid also in VS; however, the TYPE option is meaningless in the batch environment. In the VS version, the standard input is card and the standard output is PRINTER. Also in VS, if the FILE or MT option is specified, then there must be an ICESINPT DD card if the IN command, and no *filename* is given, and there must be an ICESOUT DD card if the OUT command and no *filename* is given, or generally a DD card named *filename* for any IN or OUT command.

The commands IN and OUT may be given anytime during a run and may be included in an input file as well as being typed in.

2. The *RETURN Command

*RETURN

This command tells the system to resume reading commands from the same input file which was being used when a command syntax error was detected by the command interpretor. It will cause processing to continue with the command (line) immediately following the command (line) which was in error. When a command syntax error occurs while reading from an input file, the system types out a message and returns to typewriter input mode. At this point, the user can type in the correct command and then type *RETURN on the next line to cause processing to resume from the file. Note that this command is meaningful only in CMS or an interactive environment. Also note that this will not always work for commands in repeat loops.

3. Other System Commands

MAP

This command may be used at any time to type a current map of how ICES is using memory. The output includes the name, address, and length of all modules currently in memory, the address of COMMON, and the address and length of all data pools.

VERIFY (OFF)

This command controls the command interpretor's typing back of commands and messages. If the option OFF is specified, this verification (echo typing) is suppressed. Initiation of a subsystem automatically turns verification on.

E. UTILITIES IN *CMS-ICES* FOR SUBSYSTEM DEVELOPMENT

This section describes EXEC procedures and special files for development of ICES subsystems in CMS.

1. Pre-Compilation

An EXEC file called PRECOMP will cause a specified file of type ICETRAN to be pre-compiled by the ICETRAN precompiler and the resultant output from pre-compilation to be compiled by the G1-Level Fortran compiler.

PRECOMP *filename*

where *filename* is the name of the ICETRAN source program to be ICETRAN precompiled and Fortran compiled. It must be of filetype ICETRAN and must end with **EOF starting in column 1.

The output of this procedure will be a TEXT file named *filename*. If more than one subprogram is in the source file, only one **EOF line should be in the source file (at the very end). Only one TEXT file will be produced, that file containing the objects of all the subprograms in the source file.

A listing of the ICETRAN source file and the Fortran compilation is produced on the printer.

2. Load Module Generation (CMS Version)

The EXEC procedure MAKEMOD will cause several object decks (several text files from pre-compilation) to be linkage edited together, resolving all externs among themselves and routines in TXTLIBS (such as the Fortran library), and then producing an executable module (CMS TEXT file).

MAKEMOD *subsystemname modulename*

where *subsystemname* is the name of the ICES subsystem
and *modulename* is the name of the module to be created.

This procedure assumes that two other files already exist, namely the file *subsystemname* BLDL and file *modulename* LIST (or file *subsystemname* BLDL and file *subsystemname* LIST). If *subsystemname* is not given, this procedure assumes that only one other file already exists, namely the file *modulename* LIST. These files will now be described.

a. The BLDL File

The subsystem BLDL file has two uses as described below:

1) CMS-ICES requires a file with a filetype of BLDL for each subsystem. This file records the names and sizes of the modules for the subsystem and the aliases (extra entry prints) for each module. Its filename is the

subsystem name stored in the first 8 bytes of system COMMON. The actual name depends on how the subsystem was initiated and it is suggested that the user store the subsystem name he requires in system COMMON during his restart procedure. The system will indicate if it cannot find the required subsystem BLDL file by typing the name for which it was searching. System COMMON may be preset in the restart procedure with the following CDL command:

```
PRESET ALPHA 8 'QQDUB' EQ 'subsystemname'
```

Note that QQDUB will probably have to be added to the subsystem common map at location 0.

During execution an ICES BLDL and a subsystem BLDL are read into memory.

2. This file also acts as the control for the load module generation procedure and must be set up as follows:

   Starting in column 1, the module name preceded by QQ or the alias name preceded by QQ.

   Starting in column 10, six zeros if a module name or the module name preceded by QQ if an alias name.

   Suppose a subsystem consists of 3 modules with aliases for two of those modules. Its BLDL file would be established as follows:

```
QQSUB1      000000
QQSUB2      QQSUB1
QQSUB3      000000
QQSUB4      000000
QQSUB5      QQSUB4
QQSUB6      QQSUB4
```

where the left-hand column begins in column 1 and the right-hand column begins in column 10. The module names are QQSUB1, QQSUB3, and QQSUB4, and the other names are aliases for the modules named in column 10.

b. The LIST File

The load module generation procedure requires some indication of what routines are to be included in the module and what TXTLIBs to search. The load module generation procedure requires that this information be given in a *subsystemname* or *modulename* LIST file.

Suppose a module is to be created for the TOPO subsystem. The *subsystemname* LIST file would be named TOPO LIST and would be established as follows:

```
STARTMOD
*
TOPO
ICELIB
FORTLIB
SYSLIB
```

All entries begin in column 1. STARTMOD is the routine created by the load module generation process using the TOPO BLDL as a control. It is basically a table of the entry names for the module. The * is a fence between the TEXT *filenames* and the TXTLIB *filenames*.

The LINKEDIT program (which is invoked by the MAKEMOD procedure) starts by including the STARTMOD TEXT file created for the module (created by the MAKEMOD procedure). If we were making module QQSUB1, mentioned in the BLDL example above, the LINKEDIT program would then search for the SUB1 and SUB2 in TOPO TXTLIB. Any programs referenced by SUB1 and SUB2 would be searched for also in TOPO. When no more external references can be resolved from TOPO, the next TXTLIB mentioned, namely ICELIB (the ICES system function library), would be searched. This continues until either all references have been resolved or until the end of the LIST file is reached.

If the routines to be used in the module do not exist in a TXTLIB or if updated versions are to be used which have not yet been placed in the TXTLIB, a special LIST file may be made for the module indicating exactly what routines are to be used.

Suppose that for module QQSUB1 routines SUB1 and SUB2 are not to be obtained from TOPO (i.e., they are to be obtained from TEXT files), but all other routines are to be obtained from a TXTLIB; then the SUB1 LIST file would be created as follows:

```
STARTMOD
SUB1
SUB3
*
TOPO
ICELIB
FORTLIB
SYSLIB
```

Note that STARTMOD must always be the first entry in any LIST file. The TXTLIBs mentioned are always searched in the order entered so that if a routine exists in more than one TXTLIB it will be obtained from the first TXTLIB that contained it provided that some other routine mentioned (referred to) it before or while the first TXTLIB was being searched.

c. MAKEMOD Examples

In the example above, if the *subsystemname* LIST file is to be used, the procedure for module generation would be:

MAKEMOD TOPO SUB1

This version of MAKEMOD would then do the following:

1) Execute SETGEN1 to create the STARTMOD ASSEMBLE file using the TOPO BLDL file as the control.

2) Assemble STARTMOD ASSEMBLE creating STARTMOD TEXT.

3) Execute LINKEDIT to create the module QQSUB1 using the SUB1 LIST file as a control if one exists or, if not, the TOPO LIST file as a control.

4) Execute BLDLEDIT to store the module size in the TOPO BLDL file.

5) Erase unwanted files.

If the *modulename* LIST file is to be used, the procedure for module generation would be:

MAKEMOD SUB1

This version of MAKEMOD would do the following:

1) Execute SETGEN to create the STARTMOD ASSEMBLE file using the SUB1 LIST file as the control.

2) Assemble STARTMOD creating STARTMOD TEXT.

3) Execute LINKEDIT to create the module using the SUB1 LIST file as a control.

4) Erase unwanted files.

Note that in the second example above, BLDLEDIT would not be executed, and thus the module size would not be entered into the subsystem BLDL file. This would then have to be done separately as described in the next section.

d. BLDLEDIT

BLDLEDIT is a program which inserts the module length in the appropriate position in the subsystem BLDL file. It is invoked by:

BLDLEDIT *subsystemname* QQ*modulename*

or in the second MAKEMOD example above, the MAKEMOD would be followed by:

BLDLEDIT TOPO QQSUB1

3. Load Module Generation (VS Version)

Module generation for VS consists of the following:

a. Putting any changed texts into the partitioned dataset for the subsystem object library. The VS utility program IEBUPDTE is used for this.

b. Execute PGM=QQFUBAR2 with the appropriate input for OSSETGEN to make the modules. The object library must be specified in the OBJ DD card.

The input for program OSSETGEN is as follows:

column 1	column 10
modulename	
(*textname*)	(*entryname*)
.	
.	
.	
**EOF	

where *textname* is the name in the object library of the program to be included in the load module; and *entryname* is one of the entry points (main or alias) to the module - the *modulename* entry must be the first line after the *modulename* line.

The parentheses denote optional.

4. Cross Reference Listing

A listing of all modules in a subsystem, their contents, and a list of all modules which contain each program can be produced by running program CROSS. It uses the set of input for program OSSETGEN which must be in one file. Input is on dataset 8; output is on dataset 10.

5. CDB and ICES File Generation

As in any ICES system, subsystem commands are defined to the system by inputting their definitions, using CDL commands, to the CDL subsystem. This is done as a normal ICES execution, using any of the system commands for input/output control, e.g., input from a file, output to a file, etc. The CDL subsystem creates a CDB (command definition block) for each command definition. These CDBs are written onto the ICES system file, established in CMS as DD3 ICES.

Besides DD3 ICES as the system file, the other files associated with ICES processing are:

DD1 ICES	the user data file
DD2 ICES	the subsystem data file
DD4 ICES	the dynamic memory overflow data file.

6. *Debugging in ICES*

Subsystem debugging can be simplified by using the DEBUG form of the ICES command.

ICES DEBUG QQ*modulename*

where QQ*modulename* is the name of the module in which one wishes to trace the flow of control by using breakpoints or any of the other CMS DEBUG facilities.

When module QQ*modulename* is loaded into memory, the ICES system will type out a map of memory (similar to what is produced from the MAP command) and then enter the CMS DEBUG environment. The user can then issue any valid DEBUG commands, such as breakpoints, and then use the DEBUG facility RETURN command to return to the ICES environment. After returning from the DEBUG environment, the keyboard will unlock at which time a QQ*modulename* may be typed by the user to specify the next module which he will want to investigate. If he does not want to stop again, he enters a null line. The next QQ*modulename* typed in may be the same module for which breakpoints, etc., were just specified.

III. *STRUDL* SUBSYSTEM DEVELOPMENT

The CMS-ICES system was used to develop additions and modifications to the STRUDL subsystem. These enhancements were in the finite element, dynamics, and substructuring sections. These changes were made for several reasons. First, some of these capabilities were designed into the original version of STRUDL but were not fully implemented. Second, some - especially the substructuring - needed some redesign in order to operate in a general and effective fashion. Third, changes were made to the finite element sections to include consistent loadings for most element types and to eliminate elements which were special cases of more general elements.

Details on all enhancements to STRUDL are documented elsewhere.[5]

IV. CONCLUSIONS AND RECOMMENDATIONS

The development of CMS-ICES and VS-ICES is complete and both versions are being used for production runs. The various changes made to the system and to STRUDL have resulted in substantial performance improvements, assuming judicious use of the REORG parameter.

Additional improvements could be achieved by modifying the precompiler to accept Fortran G statements and making the additional changes necessary to permit a variable block size. These additional changes were not done, due primarily to time limitations on implementation, but the time required to implement these additional changes would be on the order of 3 man-months.

An additional improvement which could be made is to develop a facility to access DD4 from both CMS and VS.

APPENDIX A

MAKING THE CMS ICES MODULE

The CMS ICES Module is made in two steps. First, LINKEDIT ICES is executed which produces an ICES TEXT file consisting of the programs listed in the ICES LINKEDIT file. There will be several unresolved externs resulting from this.

The second step is to issue:

```
GLOBAL TXTLIB FORTLIBN FORTLIB

LOAD ICES SYCONSI QQCOMADR MEMORY70 ZQBLDL ICESIO SAVAREA QQFIOSH

GENMOD ICES
```

APPENDIX B

PROGRAM MODIFICATIONS (CMS)

A. Changed Programs

1. Function library programs

a. ALOCAT, IRTEST, ISTACK, LOADNM, QQDELE, QQDEST, QQLINK, QQLNIF, QQLOAD, QQPLDP, QQPRIO, QQSTAD, QQSTDL, QQSWCH, QQSTLK, OPTION, IDAREA

Code added to store address of COMMON in register 4.

b. EXIT, QQPOOL, QQDSNC, QQINHT, QQTRAN

Code added since COMMON address is not in register 4 on entry.

c. IAVAIL

Partition size determination changed to constants.

d. QQDFIN, QQINT1, QQINT2, QQINT3, QQINT4, QQINT5, QQINT6

Modification due to COMMON address not being in register 4.

e. QQSTAT

Correction to incorrect handling when caller has levels specified which exceed lowest level of a dynamic array.

f. SYIBCOM - Csect name IBCOM

Added IHOFDUMP and DUMP as entries due to changes in Fortran G library routines.

Added STAESW as entry.

Changed parts of program since register 4 does not contain address of COMMON.

2. Secondary storage programs

a. TABLES

Changed to use CMS disk macros (FCB) rather than OS BDAM macros.

b. DISK1

Reference to INTSW changed to INTSWTCH for CMS and VS Fortran G.

Variable branch added at SPIE which is set in STRT1CES depending on whether DEBUG option is given or not.

Removed all WAIT macro code.

Changed WRITE and READ to FINIS, STATE, WRBUF/RDBUF.

c. DISK3

Changed the MVC for the ID (or TTR) of last physical record.

Added code to force writing of extended logical record.

d. DISK4

Check for whether DDNAME was specified changed to check if name was entered.

Sections to move in DDNAME into DCB modified to move name into FCB and set up FCB parameters.

Test on DISP = NEW altered.

READ section changed to RDBUF section.

Storing of blocks initialized altered.

Section added at EXIT to make sure the file directory is written out as soon as possible.

CLOSEs changed to FINIS.

Added section to force IBCOM to return to DISK4 on exit by changing SAVAREA(12) which is part of the dummy save area for Fortran G.

WAIT macros removed.

Branch to PDUMP was bypassed.

ABDUMP section removed.

Extend data set section changed to branch to error message.

INIT routine removed.

e. DSKERR

Modified to give CMS disk error numbers from RDBUF and WRBUF.

f. DISKREPA, DISKRITE

Changed to put common address into QQCOMADR and into register 4 for use by interface routines.

3. Command interpretor programs

a. COM1 - Csect COMINT$1

VERIFY, MAP, and *RETURN were added as immediate commands. The displacements in COMINT#2 DSECT were updated.

b. COM2 - Csect COMINT#2

Updated the displacements in COMINT$1 DSECT.

c. CIERS1

Calls to QQERSG and QQERFN were added to force typing of error messages.

4. Precompiler

ICPRECOM

Section added to type error messages on the terminal.

5. CDL utility programs

DCCDIS, CDLST

Changed to put address of COMMON into register 4.

6. Initiation programs

a. EXTERN

Bound changed.

COMMON section added to force a loader table entry for common when the ICES module is loaded.

b. ICEX

An instruction was added to zero register 15 upon normal completion of an ICES run.

c. INAL

Section added to store COMMON address in CMS loader tables and in QQCOMADR.

d. INIT

IOKNTRL extern and DC added for inserting its address into COMMON and the MVC for this changed accordingly.

SEGWT removed.

e. SKFUDGE - Csect name QQFUDGE

Parameter list interpretation was removed. Partition and machine size determination was changed to use NUCON tables.

7. Program and memory management programs

a. GTPOOL

DUMAR made DSECT.

b. QQIO

Altered to get COMMON address and to allow Fortran G to use register 13 as base.

Altered to permit Fortran G *end* = parameter.

c. QQRERZ

DUMVAR made DSECT.

d. QQSPC

DUMARA made DSECT

e. QQSTER

Modified to set mask bit in IOKNTRL in MEMORY to force typing of message.

f. REORP1

DUMARE made DSECT.

g. RGFAIL, QERROR

Reference to IOKNTRL added to turn typewriter on for error messages.

Call to MEMORY to give map of primary memory.

h. SKFINCH2 - Csect QQFINCH

The DELETE macro was fudged by changing all DELETE macros to branches to an added section of code called DELETE. This section issues a FREEMAIN for the space occupied by the module to be DELETED. It also removes the name from the CMS loader tables. This was necessitated by the problems with the LOAD macro.

The BLDL macro was changed to a call to the ZQBLDL routine.

The testing for and size computation of overlay modules was removed.

The LOAD macro was replaced by two sections of code, one at GOTCOR and one at LOAD macro. The first section issues a GETMAIN for the space into which the module will be placed by means of the INCLUDE command which is issued in the section at the LOAD macro.

i. SYRTPR1, DMRTSPC, QQDCT, QQSAVE

SEGWT macro removed.

8. Miscellaneous programs

a. QQSNAP

Changed to not give a dump.

b. QQSTUP

Made a dummy since not needed in CMS.

B. New Programs for CMS

1. Initiation programs

a. ICEX1

Replaces ICEX1 and TIMDAT in order to print the time and date without using a Fortran program.

b. STRTICES

This program initiates the ICES module

Checks to see if DEBUG option is specified. If so, it causes a branch around the issuance of the SPIE macro in program DISK1.

Calls STRINIT to restore any lingering free space.

Transfers to QQFUDGE.

2. Program and memory management programs

a. MEMORY70 - Csect name MEMORY

Routine to type the contents of memory in response to a MAP command or when a memory overflow occurs or when in DEBUG. The address of COMMON, address and lengths of POOLS, address, lengths, and names of modules are typed.

This routine contains a displacement in routine SKFINCH which must be changed if a change is made to FINCH. It is the displacement of TABLE in FINCH.

b. QQCOMADR

A csect to provide storage for the address of COMMON for reference by the root and other routines.

c. SYEXSTUP - Csect name EXSTUP

Replaces old OS EXSTUP program. Besides doing the standard EXSTUP functions except COMMON address storing, it also handles the DEBUG mode by checking for equivalence to debug name given, calling MEMORY for a map, reading in next DEBUG name, and issuing the CMS DEBUG command.

d. ZQBLDL - Csect name BLDL

Program to obtain module information from the file STRUDL BLDL or *subsystemname* BLDL. It replaces the BLDL macro. The module information is stored in the caller's BLIST. Call is with

```
LA   1, BLIST
L   15, =A(BLDL)
BALR 14, 15
```

Return is with the desired information in the caller's BLIST. Register 15 is set to zero if no errors and to 1 otherwise. BLIST information is set to reusable, re-entrant, nonscatter, nonoverlay. Aliases are handled properly.

BLDL also issues a GLOBAL TXTLIB command for all module txtlibs specified in the BLDL file.

This routine is arranged so as to read in an ICES BLDL containing information about the modules used by all subsystems (e.g., Command Interpretor) and a BLDL for the subsystem presently being used.

BLDL is called with register 1 pointing to a 56 byte BLIST that is the equivalent of the BLIST used by the OS BLDL macro with the module or alias name required being in the first 8 bytes. BLDL inserts in the BLIST the size of the module and if the BLIST name was an alias then the module name is also inserted. The relative entry point in the BLIST is set as zero and the module is assumed reusable, re-entrant and nonoverlay.

BLDL first checks to see if any BLDL files have already been read into its table and, if not, then the ICES BLDL is read in. Then if there is a subsystem name in the first 8 bytes of COMMON, it is checked against the subsystem name of any BLDL file residing in its tables. If the names are different the required subsystem BLDL is read into the tables.

If during the read of an ICES or *subsystemname* BLDL file the name TXTLIB is detected starting in column 1 of an entry, then the TXTLIB name given is inserted in

the parameter list for a CMS GLOBAL TXTLIB command. Once a TXTLIB has been detected all modules following this entry in the table for that BLDL file are marked as being in a TXTLIB but no distinction is made as to which TXTLIB they are in. A CMS GLOBAL TXTLIB command is issued for all the TXTLIBs detected after a BLDL file has been read in. Note that SYSLIB TXTLIB is not included in the parameter list unless specifically mentioned in a BLDL file. Note also that TXTLIBs will be searched in the order entered so that the first copy of any module found will be used and further if a TEXT file exists for the module it will be used even it is exists in a TXTLIB. The CMS dynamic loader searches TXTLIBs only if no TEXT file exists.

BLDL searches its tables for the BLIST name and checks to see if the appropriate module exists as a TEXT file unless it is marked as being in a TXTLIB.

BLDL checks for the following errors giving the message mentioned and setting register 15 to 1 to indicate an error to the calling program:

"NO *name* BLDL FILE" the required BLDL file does not exist.

"NO *name* TXTLIB FILE" the required TXTLIB does not exist.

"BLDL READ ERROR" disk read error in reading a BLDL file.

"*an entry* THIS LIST ENTRY IN ERROR" this is not a correctly formed BLDL file entry. Note that this error does not return an error code, it is simply ignored in the hope that the entry is not required.

"BLDL LIST OVERFLOW" there is not enough room in the BLDL table (400 entries) for the BLDL file or there is not enough room in the TXTLIB list (8 entries) for a TXTLIB name.

"NO LENGTH IN BLDL LIST" the module name did not have a length included in its entry.

If the BLIST name cannot be found in the BLDL table or if the module name could not be found if the BLIST name was an alias or if the module did not exist as a TEXT file and the entry did not indicate a TXTLIB, then no message is given because the ICES calling routine is responsible for giving the appropriate error message.

A correct BLDL file entry for a module name contains the name left justified in columns 1 to 8, a blank in column 9, and the size in Z6 format columns 10 to 15. The size must start with a leading zero and is the size in hex bytes.

This entry is stored in the BLDL table with a double word for the module name, a full word for the size translated to binary, and a final word containing zeroes if TXTLIB is to be searched and 'LIBE' if a TXTLIB is to be searched.

An alias entry in the BLDL file contains the name in columns 1 to 8, a blank in column 9, and the module name for which it is an alias in columns 10 to 17. It is stored in the BLDL table in two double words.

A TXTLIB entry in a BLDL file contains the name TXTLIB left justified in columns 1 to 6 and the TXTLIB name left justified in columns 10 to 17.

3. Input-output programs

a. ICESIO - Csect name IHCFIOSH

Routine to go between IBCOM and CMS to direct I/O on datasets 5 and 6 as desired. Input on dataset 5 can be from typewriter, CMS file, or tape. Output on dataset 6 can be on any combination of typewriter, printer, CMS file or tape, but at least on one. Note that disk and tape are mutually exclusive.

INDIC in common + 300 indicates to the routine whether a change from the present I/O direction is required. If the last 5 bits of INDIC are zero, the present state remains in force. Otherwise, they specify what state is required:

INDIC - Byte 4 contains the indicators:
bit 3 set indicates use of tape
bit 4 set indicates output to printer
bit 5 set indicates use of CMS file
bit 6 set indicates use of typewriter
bit 7 not set indicates an input command
bit 7 set indicates an output command

Thus the user could indicate OUTPUT DISK PRINT by setting INDIC to 13. Note that bits are numbered 0 to 7, left to right in a byte.

FNAME (8 bytes) in common + 304 indicates the file-name to be used.

FTYPE (8 bytes) in common + 312 indicates the file-type to be used.

FMODE (4 bytes) in common + 296 indicates the filemode to be used if I/O is disk file or indicates the tape number (1 for TAP1) to be used if I/O is tape. Note that tape and disk are mutually exclusive on output.

Output files on disk are added to if they exist.

The real IHOFIOSH which this routine intercepts must be obtained from the Fortran G library and arranged with the new name QQFIOSH to process the calls passed on to it by this routine.

An entry TYPE is included in ICESIO that can be used by the FORTRAN call CALL TYPE. This turns on the typewriter no matter what the I/O state currently is. There is an additional entry IOKNTRL which indicates a double word containing the I/O state indicators, the first word being for INPUT and the second for OUTPUT. These indicators are exactly as described for the COMMON indicators and are the places where ICESIO saves this information. Thus, a programmer may interrogate these words to find what the I/O state is or to change the I/O state. A number of ICES root programs use IOKNTRL to turn the typewriter on before giving an error message.

b. QQERSG - also entry QQERFN

Subroutine to turn on the typewriter bit for output for error processing. Determines the state of devices and saves them so that QQERFN can restore them after error processing is completed.

4. Miscellaneous programs

a. IHOASYNC

A dummy IHOASYNC csect to prevent unresolved externs in making modules for TOPO subsystem and STRUDL subsystem.

b. SAVAREA

Dummy section for Fortran G savearea.

c. STAESW

Dummy section for FORTRAN library reference.

d. STARTMOD

This program replaces program SETUP. It is generated by program SETGEN or SETGEN1, using file STARTMOD SAVE as a foundation and the entries in the BLDL file or the LIST file for the specific module. It defines all entries to the module, provides information for trace, and its table is used by EXSTUP to see if DEBUG should be entered.

5. Programs for module generation

a. BLDLEDIT

A new routine used in the load module generation procedure to update the entry in the BLDL file for a module. It is executed with

BLDLEDIT *subsystemname QQmodulename*

and searches the subsystemname BLDL file for the entry QQ*modulename* commencing in column 1. It then searches for a QQ*modulename* TEXT file and reads its first record. It checks that this is an ESD record containing the name QQ*modulename* and moves the three byte module size to columns 10 to 15 of the BLDL entry.

b. LINKEDIT

The LINKEDIT module creates dynamic MODULEs from TEXT files. The created MODULEs are normal relocatable CMS TEXT files with all internal references resolved and unreferenceable by other routines. The MODULEs are entered at relative location 0, with the entry name being the MODULE name. The TEXT file created has the MODULE length included on the first ESD card so that it may be used correctly by the CMS dynamic loading routines. The MODULE may have unresolved external references to other routines that will be resolved at load time. The length for each final COMMON name is the largest length for that COMMON name, but this length is ignored by the dynamic loader.

The program is invoked with the following call:

LINKEDIT *modname filenames (options) txtlibnames*

where:

modname	is the name to be given to the LINKEDITed file created.
filenames	are the names of the TEXT files to be included in the MODULE.
options	are TYPE, NOTYPE, PRINT, and NOPRINT.
txtlibnames	are the TXTLIBS to be searched for unresolved references.

An existing TEXT file with the name, *modname*, will be erased before the LINKEDITed file is created.

If an input file does not exist, a message will be typed and processing continued ignoring it. It is permissible for one of the TEXT files to be included to have the same name as the TEXT module to be created,

but it must be realized that it will be replaced by the LINKEDITed module. Note that because the TEXT module is entered at relative location 0, its execution will begin with the first routine to be included.

Alternatively, if no filenames are specified, the names will be obtained from the file *modname* LINKEDIT. This file must consist of 80 character records with the first eight characters of each being a *filename*. Records starting with a blank in column 1 are ignored, and a record starting with an * in column 1 terminates the obtaining of *filenames* from it.

If unresolved external references exist after LINKEDITing the TEXT files specifically mentioned in the parameter list or the LINKEDIT file, further routines are obtained from the TXTLIBs mentioned in the parameter list. These TXTLIBs are searched in the order entered. If a LINKEDIT file is being used, the TXTLIBs to be searched must be included in the LINKEDIT file as for the TEXT files, but after the * record mentioned above.

Although any combination of the *options* is allowable, the last of associated *options* will be used. The default options are TYPE and NOPRINT.

The possible typed and printed output contains:

a. module name
b. list of TEXT files included
c. external references left after TEXT files used
d. TXTLIBs searched
e. size of the MODULE

The TYPE *option* types all the items specified above. The PRINT *option* causes all the items mentioned to be output on the off-line printer. NOPRINT is obvious. The *option* NOTYPE inhibits the typing of (b), (c), (d), and (e), but error messages and a list of the final unresolved external references in the MODULE will be typed.

c. SETGEN

This program creates STARTMOD ASSEMBLE from STARTMOD SAVE using the *modulename* LIST file to indicate the entry names for the module. The entry names are found in the LIST file starting in column 10. They are included in STARTMOD with QQ preceding them.

d. SETGEN1

A similar program to SETGEN but *subsystemname* BLDL is searched to find the entry names. This program is called with

SETGEN1 *subsystemname modulename*

The *subsystemname* BLDL file is searched for *modulename* starting in column 3. When found it is checked for consistency in that there must be a QQ in columns 1 and 2, column 9 must be blank, and column 10 must contain a zero. Only such entries are accepted as correct module entries. The module name and following alias names are placed in the STARTMOD ASSEMBLE file being created. Alias names must be in entries immediately following the module name entry and must contain QQ in columns 1 and 2, a blank in column 9, and QQ*modulename* in columns 10 to 17. Anything else is treated as an error.

C. Summary of New and Changed Programs

CHANGED			NEW
ALOCAT	INIT	QQLNIF	BLDLEDIT
CDLST	IRTEST	QQLOAD	ICESIO
CIERS1	ISTACK	QQPLDP	ICEX1
COM1	LOADNM	QQPOOL	IHOASYNC
COM2	OPTION	QQPRIO	LINKEDIT
DCCDIS	PRECOM	QQRERZ	MEMORY70
DISKREPA	QERROR	QQSAVE	QQCOMADR
DISKRITE	QQDCT	QQSNAP	QQERSG
DISK1	QQDELE	QQSPC	SAVAREA
DISK3	QQDEST	QQSTAD	SETGEN
DISK4	QQDFIN	QQSTAT	SETGEN1
DSKERR	QQDSNC	QQSTDL	STAESW
EXIT	QQINHT	QQSTER	STARTMOD
EXTERN	QQINT1	QQSTLK	STRTICES
FINCH2	QQINT2	QQSTUP	SYEXSTUP
FUDGE	QQINT3	QQSWCH	ZQBLDL
GTPOOL	QQINT4	QQTRAN	
IAVAIL	QQINT5	REORP1	
IBCOM	QQINT6	RGFAIL	
ICEX	QQIO	RTPR1	
IDAREA	QQLINK	RTSPC	
INAL		TABLES	

APPENDIX C
PROGRAM MODIFICATIONS (VS)

A. Changed Programs

1. Secondary storage programs

a. OSTABLES

Macros changed to BDAM.

Constants changed for block size of 4000.

b. OSDSKBFS, OSSAB

Changed for block size of 4000.

c. OSDISK1

Restored all WAIT macro code.

Changed CMS macros back to BDAM macros

Changed constant for 4000 block size.

d. DISK2, DISK6, DISK7

Reassembled with VS version of DISKDATA macro generation.

e. OSDISK3

Removed changes made to original OS version for CMS version.

f. OSDISK4

CMS changes were restored to original form except for section added to force IBCOM to return to DISK4 which was left the same as the CMS addition.

Constants changed for block size of 4000.

g. OSDISK5

Instruction changed for block size of 4000.

2. Precompiler

OSPRECOM

Removed section to type error messages on the terminal.

3. Initiation programs

a. OSEXTERN

COM section removed.

b. OSFUDGE

Restored CMS removals.

c. OSICEX1

Restored to original form.

d. OSINAL

References to CMS loader table removed.

e. OSTIMDAT

Changed so does not type but returns information to be printed.

4. Program and memory management programs

a. OSFINCH2

Changes for DELETE macro were restored to original.

BLDL macro was restored.

Overlay computation was restored.

LOAD macro changes were restored to original.

Note again that any change to OSFINCH2 must be reflected in a change to the table displacement constant in OSMEM70 at TABDISP.

b. OSGTPOOL

Changed so that contiguous pools will not be made into one pool.

c. OSMEM70 - Csect name MEMORY

Output operations changed to OS BSAM macros. Output is printed on DD*name* FT06F003.

5. Input-output programs

OSICESIO - Csect name IHCFIOSH

I/O Operations changed to OS BSAM macros.

The typewriter mode for output is meaningless in VS.

Standard input is from DD*name* FT05F001 (corresponds to CMS TYPE input).

Alternate default input is from DD*name* ICESINPT (corresponds to CMS FILE input).

Standard output is on DD*name* FT06F001 (corresponds to CMS PRINTER output). Alternate default output is on DD*name* ICESOUT (corresponds to CMS FILE output).

Input and output to TAPE is meaningless in VS since the DD card defines the medium and the macros are independent of device (disk or tape) for sequential files. Switching can still be done (as in CMS) but between DD*name*s FT05F001 and ICESINPT or *filename*, or between FT06F001 and ICESOUT or *filename*. Note that *filename* here is the name given in the IN or OUT command.

6. Module generation programs

FUBAR2

Removed overlay check.

Bypass ID = -1 check after SETGEN.

7. Miscellaneous programs

a. OSQQSNAP

Restored to its original OS form.

b. QQQQINIT

VS only. Changed from OS version to permit block size of 4000.

B. New Programs for VS

1. Module generation programs

a. ADTAB

Called by program OSSETGEN to find all adcons to COMMON in a specified object program and return these adcons to program OSSETGEN. The object program must be a member of a partitioned dataset defined on a DD card with DD*name* OBJ.

b. NIXCO

This program changes the length of COMMON in a TEXT file to 4. The name of the file is read from dataset number 5. This program is invoked in the PRECOMP procedure after compilation. It calls program NIXCO1.

c. NIXCO1

Used with NIXCO in changing the length of the ESD card for COMMON in a TEXT file to a length of 4.

d. OSSETGEN

Creates the SETUP program for VS load modules. SETUP is the entry point and starting program for all VS load modules in an ICES subsystem. It also generates the necessary input cards for the linkage editor for generating a load module for a subsystem. It calls

program ADTAB to find all adcons to COMMON in each subprogram to be included in a load module. Appropriate code is then generated, to be a part of the generated SETUP program, to resolve all such adcons to the correct subsystem COMMON address each time the module is loaded during a subsystem execution.

e. SETUP

Replaces the CMS STARTMOD program. It performs all the functions that the STARTMOD program performed. It also takes care of the common resolution function. This is performed by code and table generated by programs OSSETGEN and ADTAB. This code goes through the table, which consists of addresses of adcons for COMMON, gets the relocated value for that adcon, subtracts the relocation constant (the address of link-edited COMMON) from the value, adds the actual address of COMMON to the value, and stores the new value back in the location of the adcon.

2. *Miscellaneous programs*

CROSS

Produces a cross reference listing of programs in an ICES subsystem. Input to this program, which runs on CMS, comes from dataset number 8 and is the set of VS SETGEN input records for that subsystem. Output is directed to dataset number 10.

C. Summary of New and Changed Programs

CHANGED	NEW
DISK1	ADTAB
DISK2	CROSS
DISK3	NIXCO
DISK4	NIXCO1
DISK5	OSSETGEN
DISK6	SETUP
DISK7	
DSKBFS	
EXTERN	
FINCH2	
FUBAR2	
FUDGE	
GTPOOL	
ICESIO	
ICEX1	
INAL	
MEMORY70	
PRECOM	
QQQQINIT	
QQSNAP	
SAB	
TABLES	
TIMDAT	

APPENDIX D

SUMMARY OF *ICES* RELATED CMS FILES

EXEC files

+ PRECOMP
+ MAKEMOD
+ EXECICE

MODULE files

ICETRAN
SETGEN
SETGEN1
LINKEDIT
BLDLEDIT
ICES
CROSS

BLDL files

ICES
QQCDLCDP
STRUDL
TOPO

MACLIB files

ICESLIB
OSICELIB

TXTLIB files

CDLMODS
ICESMODS
ICELIB

ICES files

DD3
DD2
DDMASTR

Other File

+ STARTMOD SAVE

BSOSJOB OSJOBLIB file

+ BSSTROBJ
+ BSSTRSET
+ BSPRECOM
+ BSSTRUDL
+ BSSTURES

+ Listing of these files can be found in Appendix E.

APPENDIX E

LISTINGS OF SOME FILES AND PROCEDURES

STARTMOD SAVE

PRECOMP EXEC

MAKEMOD EXEC

EXECICE EXEC

BSSTROBJ OSJOB

BSSTRSET OSJOB

BSPRECOM OSJOB

BSSTRUDL OSJOB

BSSTURES OSJOB

REFERENCES

1. Schumacker, B., "An Introduction to ICES," Research Report R67-47, Civil Engineering Systems Laboratory, M.I.T., Cambridge, Massachusetts (September 1967).

2. Nilsson, R., "CMS-ICES," Urban Systems Laboratory, M.I.T., Cambridge, Massachusetts (August 1969), unpublished report.

3. Nilsson, R., "CMS-ICES Program Logic," Urban Systems Laboratory, M.I.T., Cambridge, Massachusetts (August 1969), unpublished report.

4. Logcher, R. D. and Folinus, J. J., "Interactive ICES Under TSO," Research Report R73-10, Civil Engineering Department, M.I.T., Cambridge, Massachusetts (January 1973).

5. Britten, S. S., "STRUDL Finite Elements," Lincoln Laboratory, M.I.T., Lexington, Massachusetts (April 1976), unpublished report.

FILE: STARTMOD SAVE

```
*                                                                       STA00010
STARTMOD CSECT                                                          STA00020
         ENTRY QQCOMADR                                                 STA00030
         ENTRY QQSETUP,QQFINCHL,QQSAVEAR,QQTABLE,INTSW,QQLODADR         STA00040
QQLODADR EQU   *                                                        STA00050
QQSETUP  STM   14,12,12(13)                                             STA00060
         BALR  12,0                                                     STA00070
         USING *,12                                                     STA00080
         LA    5,QQSAVEAR                                               STA00090
         ST    13,4(0,5)                                                STA00100
         ST    5,8(0,13)                                                STA00110
         LR    13,5                                                     STA00120
         L     2,0(0,1)                                                 STA00130
         MVC   QQPROGNM(8),0(2)                                         STA00140
         L     4,4(0,1)                                                 STA00150
         ST    4,QQCOMADR                                               STA00160
         LA    15,INTSW                                                 STA00170
         ST    15,INTSWLOC(4)                                           STA00180
         L     15,QQSTPAD(4)                                            STA00190
         BALR  14,15                                                    STA00200
INTSW    DC    AL1(0)                                                   STA00210
QQSAVEAR DS    18F                                                      STA00220
QQTRACE  DS    4F                                                       STA00230
QQCOMADR DS    1F                                                       STA00240
QQTBADR  DC    A(QQCOUNTS)                                              STA00250
QQPROGNM DS    2F                                                       STA00260
QQFINCHL DC    A(QQPROGNM)                                              STA00270
QQLODPT  DC    A(QQLODADR)                                              STA00280
QQSTPAD  EQU   264                                                      STA00290
INTSWLOC EQU   284                                                      STA00300
QQCOUNTS DC    AL4(TABLEND)                                             STA00310
QQTABLE  EQU   *                                                        STA00330
```

FILE: PRECOMP EXEC

```
&CONTROL OFF NOMSG
STATE &1 ICETRAN
&IF &RETCODE NE 0 &GOTO -NOFILE
ERASE &1 FORTRAN
ERASE ICETRAN SCRATCH
&ERROR &GOTO -CLEAN
FILEDEF FILE1 DISK &1 ICETRAN A1 (RECFM FS BLOCK 80)
FILEDEF FILE2 DUMMY (RECFM FS BLOCK 80)
FILEDEF FILE3 PRINTER
FILEDEF FILE4 DISK &1 FORTRAN A1 (RECFM FS BLOCK 80)
FILEDEF FILE5 DISK ICETRAN SCRATCH A1 (RECFM FS BLOCK 1600)
CP SPOOL PRINTER CONT
LOADMOD ICETRAN
START ICETRAN
FORTGI &1 (PRINT)
ERASE &1 FORTRAN
CP SPOOL PRINTER NOHOLD NOCONT
ERASE ICETRAN SCRATCH
CP CLOSE PRINTER
&X = &1
&STACK &X
NIXCO
&EXIT 0
-CLEAN &CONTINUE
&ERROR &CONTINUE
ERASE &1 FORTRAN
CP SPOOL PRINTER NOHOLD NOCONT
ERASE ICETRAN SCRATCH
CP CLOSE PRINTER
&EXIT 10
-NOFILE &EXIT 99
ERASE &1 FORTRAN
```

FILE: MAKEMOD EXEC

```
&ERROR &CONTINUE
&CONTROL ERROR
&INDEX1 = 1
&IF &INDEX GT 1 &GOTO -BLDL
&ARGS &1 &1
&TYPE SETGEN
SETGEN &1
&IF &RETCODE NE 0 &GOTO -END
&GOTO -ASS
-BLDL &SYS = &1
&INDEX1 = 2
&TYPE SETGEN1
SETGEN1 &1 &2
&IF &RETCODE NE 0 &GOTO -END
&CONTROL OFF NOMSG
STATE &2 LIST *
&IF &RETCODE NE 0 &GOTO -LIST
&ARGS &2 &2
&GOTO -ASS
-LIST STATE &1 LIST *
&IF &RETCODE EQ 0 &GOTO -ASS
&TYPE NO &1 LIST FILE
&GOTO -END
-ASS &CONTINUE
&CONTROL ERROR
&TYPE ASSEMBLE
GLOBAL MACLIB ICESLIB SYSLIB
ASSEMBLE STARTMOD (NOLIST)
&IF &RETCODE NE 0 &GOTO -END
COPY &1 LIST * QQ&2 LINKEDIT A1
&TYPE LINKEDIT
LINKEDIT QQ&2 (MAP PRINT)
&IF &RETCODE NE 0 &INDEX1 = 1
ERASE QQ&2 LINKEDIT
&IF &INDEX1 LT 2 &GOTO -END
&TYPE BLDLEDIT
BLDLEDIT &SYS QQ&2
-END &CONTINUE
&CONTROL OFF NOMSG
ERASE STARTMOD ASSEMBLE
ERASE STARTMOD TEXT
CLOSE PRINTER
PRINT LINKEDIT MAP
&EXIT
```

FILE: EXECICE EXEC

```
&FN = ZIP
&IF &INDEX NE 2 &GOTO -STOR
&FN = &1
&FT = &2
-STOR &TYPE IS STORAGE DEFINED AS 1024K ??
&READ ARGS
&IF &1 NE YES &GOTO -DSTOR
&IF &FN NE ZIP &GOTO -TDSK
-NME &TYPE ENTER INPUT FILENAME FILETYPE
&READ ARGS
&IF &INDEX NE 2 &GOTO -NME
&FN = &1
&FT = &2
-TDSK EXEC TDISK
&CONTROL ERROR
ERASE DD4 ICES D1
ERASE &FN OUTPUT D1
ERASE ICES OUTPUT
UNSHARE
&ERROR &EXIT
SHARE BLIVET B2
SHARE FANNING C2
COPY DDMASTR ICES C2 DD4 ICES D1
UNSHARE FANNING
&A = '
&X = &CONCAT &A &FN &A
&Y = &CONCAT &A &FT &A
&STACK ICES
&STACK OUT DISK   &X 'OUTPUT' 'D1'
&STACK IN DISK &X &Y
&STACK FIN
&END
&EXIT
-DSTOR &TYPE DEFINE STORAGE AS 1024K , RE-IPL, AND RE-START
&EXIT
```

FILE: BSSTROBJ OSJOB

```
// 'SCHUMACKER 0201 73',TIME=3,CLASS=B,CPU=L
//  EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=ICES.OBJ.NEWSTRUD,DISP=OLD,UNIT=PDISK,
//  VOL=SER=D67,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//  SPACE=(CYL,(35,2,200))
//SYSUT1 DD DSN=ICES.OBJ.NEWSTRUD,DISP=OLD,UNIT=PDISK,
//  VOL=SER=D67,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN DD *
```

FILE: BSSTRSET OSJOB

```
//   'SCHUMACKER 0201 73',TIME=3,CLASS=B,CPU=L
//SETUP EXEC PGM=QQFUBAR2
//STEPLIB DD DSN=ICES.NEWLINK,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//FT06F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133)
//FT07F001 DD SYSOUT=B
//FT02F001 DD UNIT=TDISK,SPACE=(TRK,(10,10)),DISP=(NEW,PASS),
//   DCB=(RECFM=F,BLKSIZE=80)
//SYSPUNCH DD UNIT=PDISK,DISP=(NEW,PASS),SPACE=(80,(25,25)),
//   DCB=(RECFM=F,BLKSIZE=80),VOL=SER=D67
//FT04F001 DD UNIT=PDISK,SPACE=(80,(25,25)),DISP=(NEW,PASS),
//   DCB=(RECFM=F,BLKSIZE=80),VOL=SER=D67
//SYSUT2 DD UNIT=TDISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3520
//SYSUT3 DD UNIT=TDISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3520
//MACLIB DD DSN=ICES.NEWMAC,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//SYSLIB DD DSN=ICES.NEWFUNC,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//   DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSLMOD DD DSN=ICES.MODULES.NEWSTRUD,UNIT=PDISK,DISP=OLD,
//   VOL=SER=D67
//OBJ DD DSN=ICES.OBJ.NEWSTRUD,DISP=OLD,UNIT=PDISK,VOL=SER=D67
//SYSUT1 DD UNIT=TDISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3520
//SYSLIN DD DSN=*.SYSPUNCH,DCB=(RECFM=F,BLKSIZE=80),
//   VOLUME=REF=*.SYSPUNCH,DISP=(OLD,PASS)
//   DD DSN=*.FT04F001,
//   VOLUME=REF=*.FT04F001,DISP=(OLD,PASS)
//FT05F001 DD UNIT=TDISK,SPACE=(80,(50,50))
//SYSIN DD *
```

FILE: BSPRECOM OSJOB

```
//ICETRAN EXEC PGM=QQFUBAR,PARM=ST
//STEPLIB DD DSN=ICES.NEWLINK,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//FILE2 DD DUMMY
//FILE3 DD SYSOUT=A
//FILE4 DD UNIT=TDISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3520
//FILE5 DD UNIT=TDISK,SPACE=(CYL,(1,1)),DCB=BLKSIZE=3520
//SYSTERM DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=TDISK,SPACE=(TRK,(10,10)),DCB=BLKSIZE=3520
//SYSUT2 DD UNIT=TDISK,SPACE=(TRK,(10,10)),DCB=BLKSIZE=3520
//OUTPUT DD DISP=(MOD,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),
//   UNIT=TDISK,SPACE=(TRK,(10,10,20))
//SYSLIN DD UNIT=TDISK,SPACE=(TRK,(10,10)),DCB=(RECFM=FB,LRECL=80,
//   BLKSIZE=800),DISP=(,PASS)
//FILE1 DD *,DCB=BLKSIZE=80
/*
```

FILE: BSSTRUDL OSJOB

```
//  'SCHUMACKER 0201 73',TIME=3,CLASS=D,CPU=L
//ICETST  EXEC  PGM=QQQICEX5,PARM=1500
//STEPLIB  DD  DSNAME=ICES.NEWLINK,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//  DD DSN=ICES.MODULES.NEWSTRUD,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//FT06F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133)
//ICESOUT DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133)
//FT06F002 DD SYSOUT=A
//FT06F003 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
//DD1 DD UNIT=TDISK,SPACE=(4000,(10,1)),DCB=(DSORG=DA,BLKSIZE=4000),
//             DSNAME=&STUDL1
//DD2 DD DSN=ICES.DATA.NEWSTRUD,DISP=OLD,UNIT=PDISK,
//  VOL=SER=D67,DCB=(DSORG=DA,BLKSIZE=4000)
//DD3 DD DSN=ICES.CDBS.NEW,DISP=OLD,UNIT=PDISK,
//  VOL=SER=D67,DCB=(DSORG=DA,BLKSIZE=4000)
//DD4 DD UNIT=TDISK,SPACE=(4000,(1200,100)),
//             DCB=(DSORG=DA,BLKSIZE=4000),
//             DSNAME=&STUDL3
//SYSUDUMP DD SYSOUT=A
//FT05F001 DD DDNAME=SYSIN
//SYSIN DD *
```

```
FILE: BSSTURES OSJOB


// 'SCHUMACKER 0201 73',TIME=3,CLASS=B,CPU=L
//ICETST  EXEC  PGM=QQQICEX5,PARM=1500
//STEPLIB  DD  DSNAME=ICES.NEWLINK,DISP=SHR,UNIT=PDISK,VOL=SER=D67
//FT06F001 DD SYSOUT=A,DCB=(RECFM=UA,BLKSIZE=133)
//FT06F002 DD SYSOUT=A
//FT07F001 DD SYSOUT=B
//DD1 DD UNIT=TDISK,SPACE=(4000,(10,1)),DCB=(DSORG=DA,BLKSIZE=4000),
//              DSNAME=&STUDL1
//DD2 DD UNIT=TDISK,SPACE=(4000,(10,1)),DCB=(DSORG=DA,BLKSIZE=4000),
//  DSNAME=&STUDL2
//DD3 DD DSN=ICES.CDBS.NEW,DISP=OLD,UNIT=PDISK,
//  VOL=SER=D67,DCB=(DSORG=DA,BLKSIZE=4000)
//DD4 DD UNIT=TDISK,SPACE=(4000,(10,1)),
//              DCB=(DSORG=DA,BLKSIZE=4000),
//              DSNAME=&STUDL3
//SYSUDUMP DD SYSOUT=A
//FT05F001 DD DDNAME=SYSIN
//SYSIN DD *
CDL
SYSTEM 'STRUDL ' 'RDL'
COMMON
'INDIC' 300
END COMMON
RESTART 'STRUDL ' 'RDL'
SIZE OF COMMON 3500
SIZE OF POOL 81920
MODULE LINK
PRESET ALPHA 8 'QQDUB' EQUAL 'STRUDL '
PRESET INTEGER 'INDIC' EQ 9
MESSAGE ' '
MESSAGE '            ************************************************'
MESSAGE '            *                                              *'
MESSAGE '            *              ICES STRUDL-III                 *'
MESSAGE '            *      THE STRUCTURAL DESIGN LANGUAGE          *'
MESSAGE '            *                                              *'
MESSAGE '            *             LINCOLN LABORATORY               *'
MESSAGE '            *   MASSACHUSETTS INSTITUTE OF TECHNOLOGY      *'
MESSAGE '            *         CAMBRIDGE, MASSACHUSETTS             *'
MESSAGE '            *          V3 M0   FEBRUARY, 1976              *'
EXECUTE 'STIME'
MESSAGE '            *                                              *'
MESSAGE '            ************************************************'
$ LOOK FOR STRUDL RESTORE 'ID'
DATA CHECK SET 'IK'
CONDITION 'IK' EQ 0
  CALL 'MESSAGE'
  NEW COMMAND
OTHERWISE
NO ID ALPHA 8 'JOBID' STANDARD 'NONE'
NO ID ALPHA 64 'TS' STANDARD 'NONE GIVEN'
ID 'POOL' INTEGER 'I1' STANDARD 0
ID 'MESS' ALPHA 4 'I2' STANDARD 'ON  '
CONDITION ALPHA 4 'I2' EQ 'ON  '
  CALL 'MESSAGE'
END CONDITION OPTIONAL
PRESET 'IO' EQ 6
EXECUTE 'STINMD'
END CONDITION
FILE
FINISH
/*
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-77-27	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Development of a New ICES Executive for the IBM/370 CMS and VS Operating Systems		5. TYPE OF REPORT & PERIOD COVERED Technical Note 6. PERFORMING ORG. REPORT NUMBER Technical Note 1977-1
7. AUTHOR(s) Betsy Schumacker		8. CONTRACT OR GRANT NUMBER(s) F19628-76-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Element No. 63431F Project No. 1227
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Systems Command, USAF Andrews AFB Washington, DC 20331		12. REPORT DATE 11 January 1977 13. NUMBER OF PAGES 70
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB Bedford, MA 01731		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

ICES executive
IBM/370
computing environments
Conversational Monitor System (CMS)
Virtual Memory System (VS)

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A new ICES executive was developed for the IBM/370 to provide more flexibility for engineering use via on-line and background computing environments and also to provide ease of maintainability and improve further development. Input can be prepared and checked (syntactically and graphically) in an on-line mode, small analyses run and checked on-line, large analyses run in batch mode, and results checked graphically on-line.

Two new versions of ICES exist: an on-line (CMS) version and a batch (VS) version. The source, except for a few programs, is independent of the version. This report describes the changes made, the reasons for the changes, new commands developed for on-line use, and performance comparisons with the previous M.I.T. version of ICES.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*